Music Information Retrieval with Neural Nets

W210 Capstone Project, Week 10

Madeleine Bulkow | Kuangwei Huang | Weixing Sun

Agenda

1. Background and Opportunities

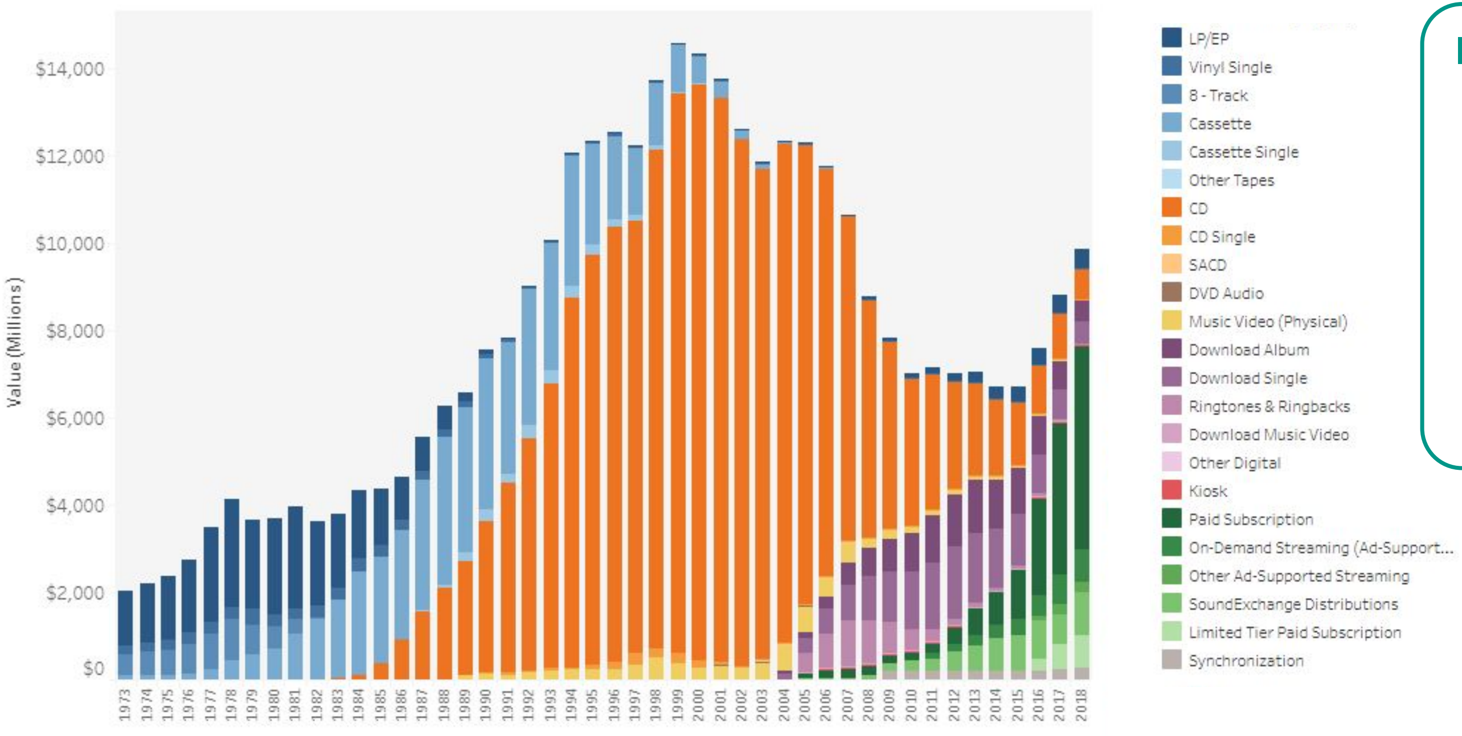
2. Data Extraction
   a. Waveform vs. Spectrogram
   b. librosa Library Usage
   c. Sub-sampling of Data

3. 2D Convolutional Neural Net
   a. Simple CNN
   b. Transfer Learning

4. Next Steps

Background: Music Industry

U.S. Recorded Music Revenues by Format 1973 to 2018



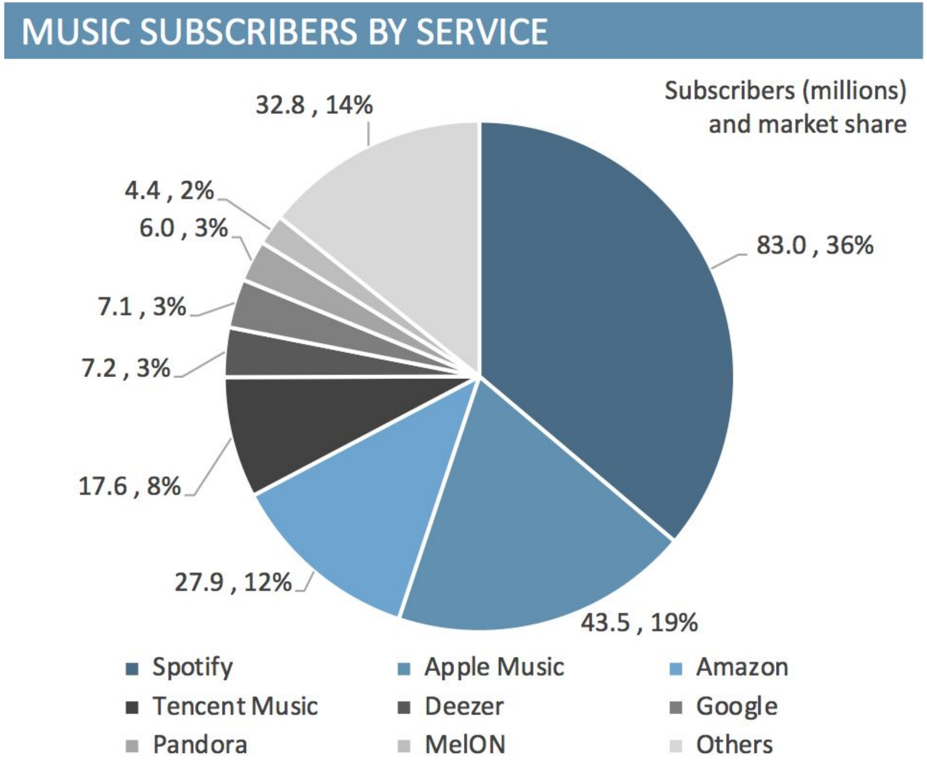
In 2018:

- $9.2 billion
- 12% YoY growth
- 75% streaming
- > 50 mil paid subscriptions

Source: https://www.riaa.com/u-s-sales-database/

Streaming Music: Opportunities

- Globally streaming revenues of $8.9B in 2018
- Streaming revenues grew by 34.0% in 2018
- Platforms compete on personalized content and "discovery"
- Recommender systems, traditionally content-agnostic
- Opportunity for content-based recommendation using deep learning
- Song profiling, akin to NLP word embeddings
- Future: GANs for music generation

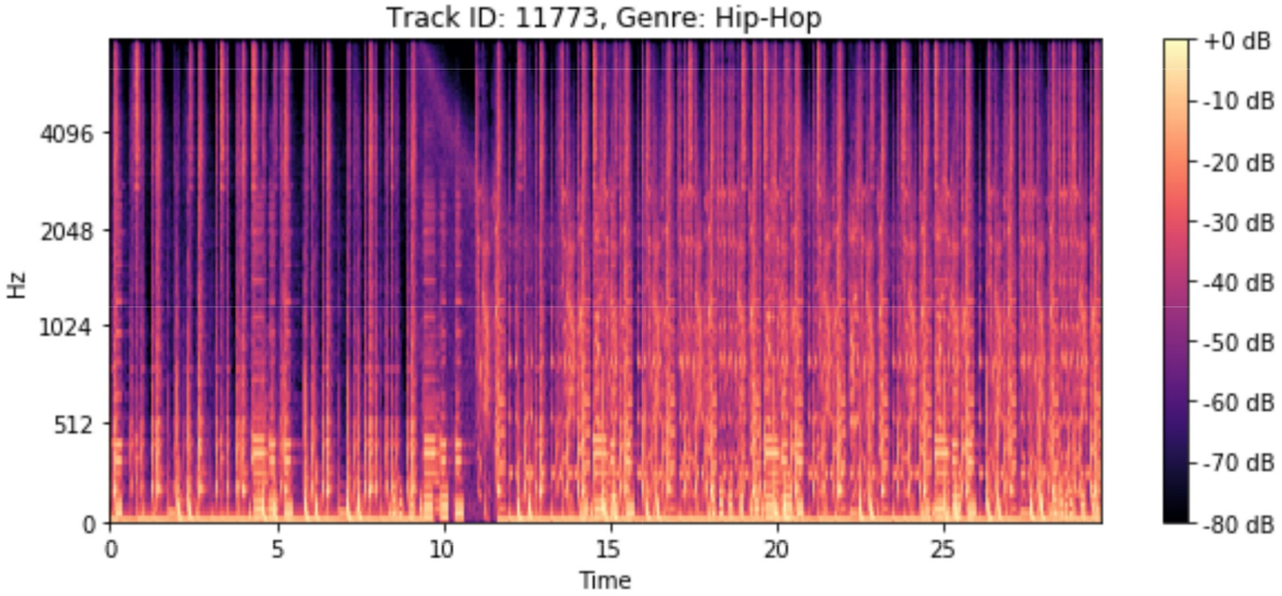


Source: https://www.ifpi.org/news/IFPI-GLOBAL-MUSIC-REPORT-2019
Source: https://www.midiaresearch.com/app/uploads/2018/09/midia-mid-year-2018-subscriber-mareket-shares.png

Data Extraction

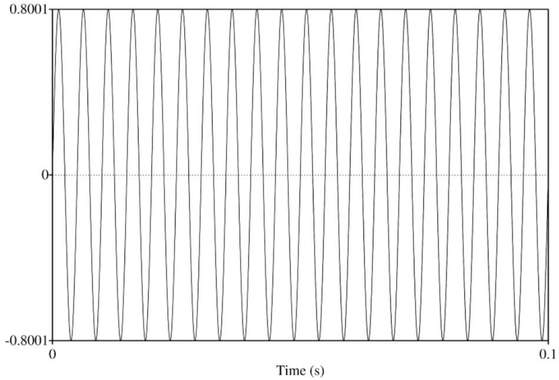
Main library: librosa
Function: convert time-series audio signals to spectrograms
Goal: visualize music patterns for genre classifications

Spectrogram: Frequency map with decibel intensity over the time duration



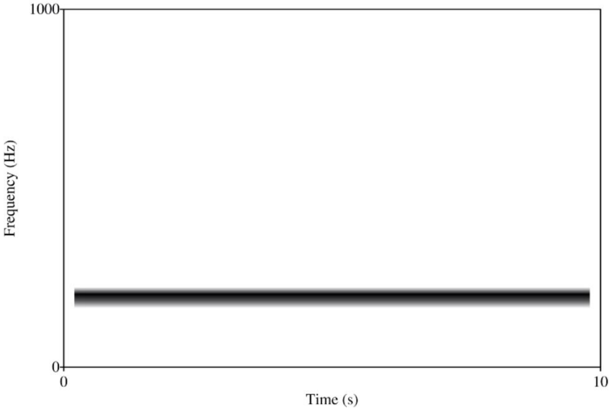
Waveform vs. Spectrogram

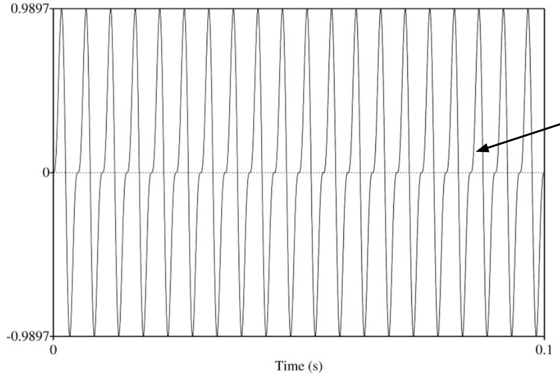
Wave



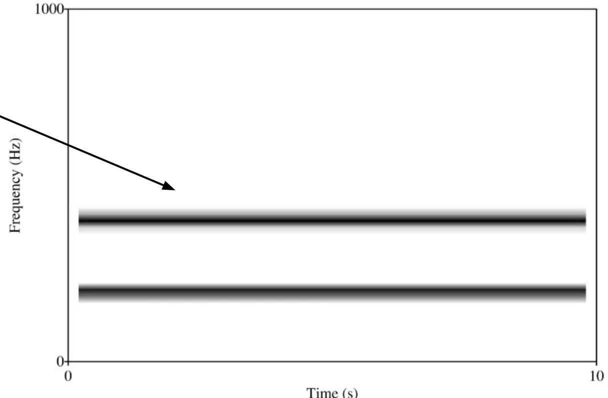
Fast Fourier Transform (FFT)

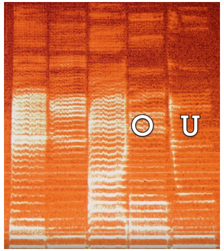
Spectrogram





Overtune + FFT

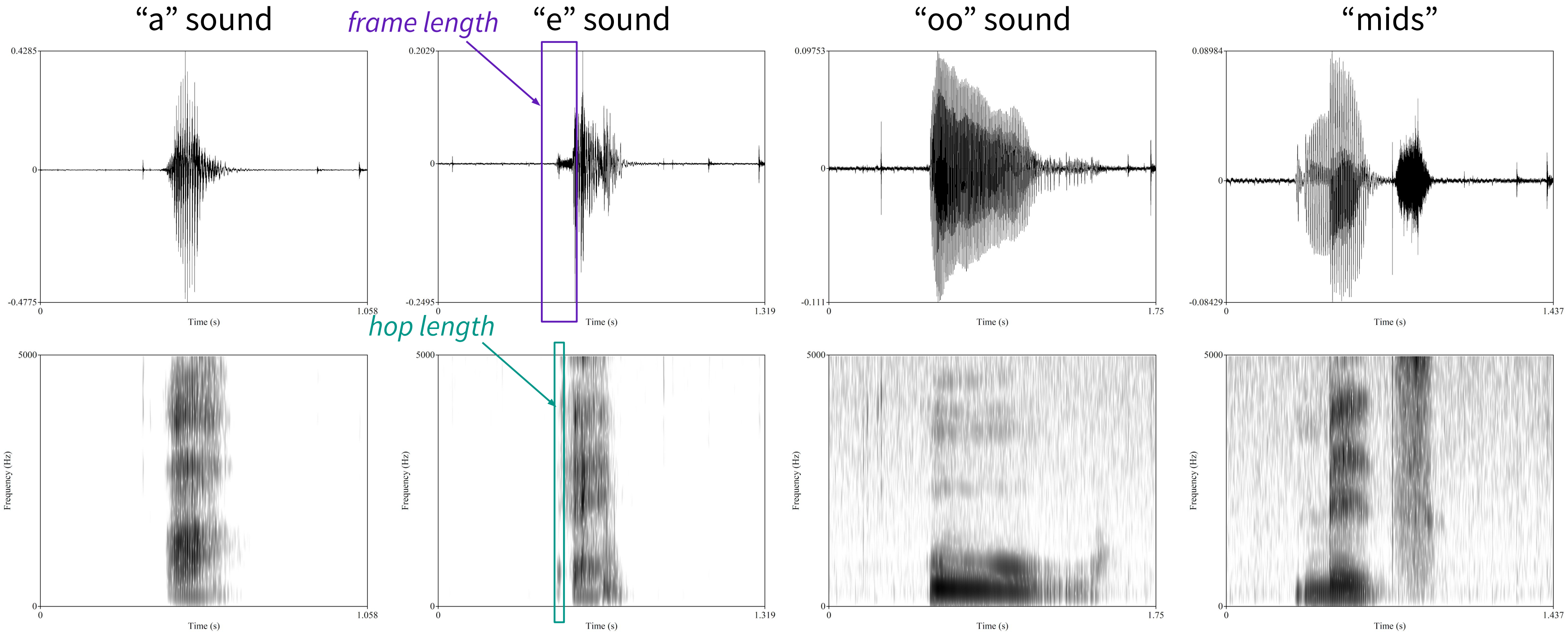




Waveform vs. Spectrogram

http://www.fon.hum.uva.nl/praat/

Doing our own recordings...



librosa Library Usage

 # Obtain the waveform "y" in time-axis and the sample rate "sr"
y, sr = librosa.load(filepath)
print("(Time series 'y', Sample rate 'sr'): ({},{})".format(len(y),sr))
>> (Time series 'y', Sample rate 'sr'): (660984,22050)

default
- *all audio is mixed to mono*
- *resampled to 22,050 Hz*
- *30s audio ~ 661,500 length*

 # Generate mel spectrograms and convert dB scale
spect = librosa.feature.melspectrogram(y=y, sr=sr, n_fft=2048, hop_length=512)

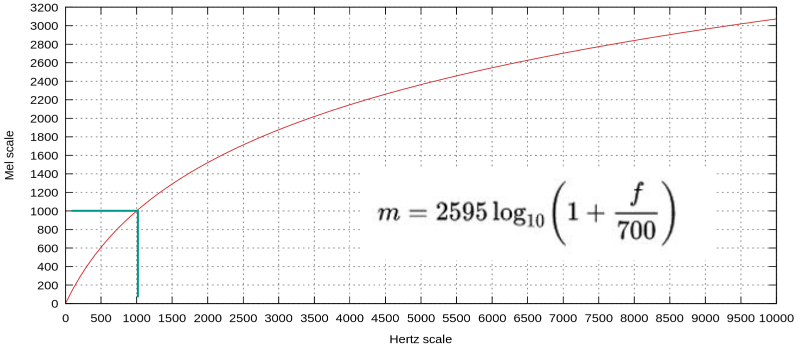
n_fft -> frame length:
The number of samples in an analysis window (or frame).

hop length -> the columns of a spectrogram
The number of samples between successive frames.

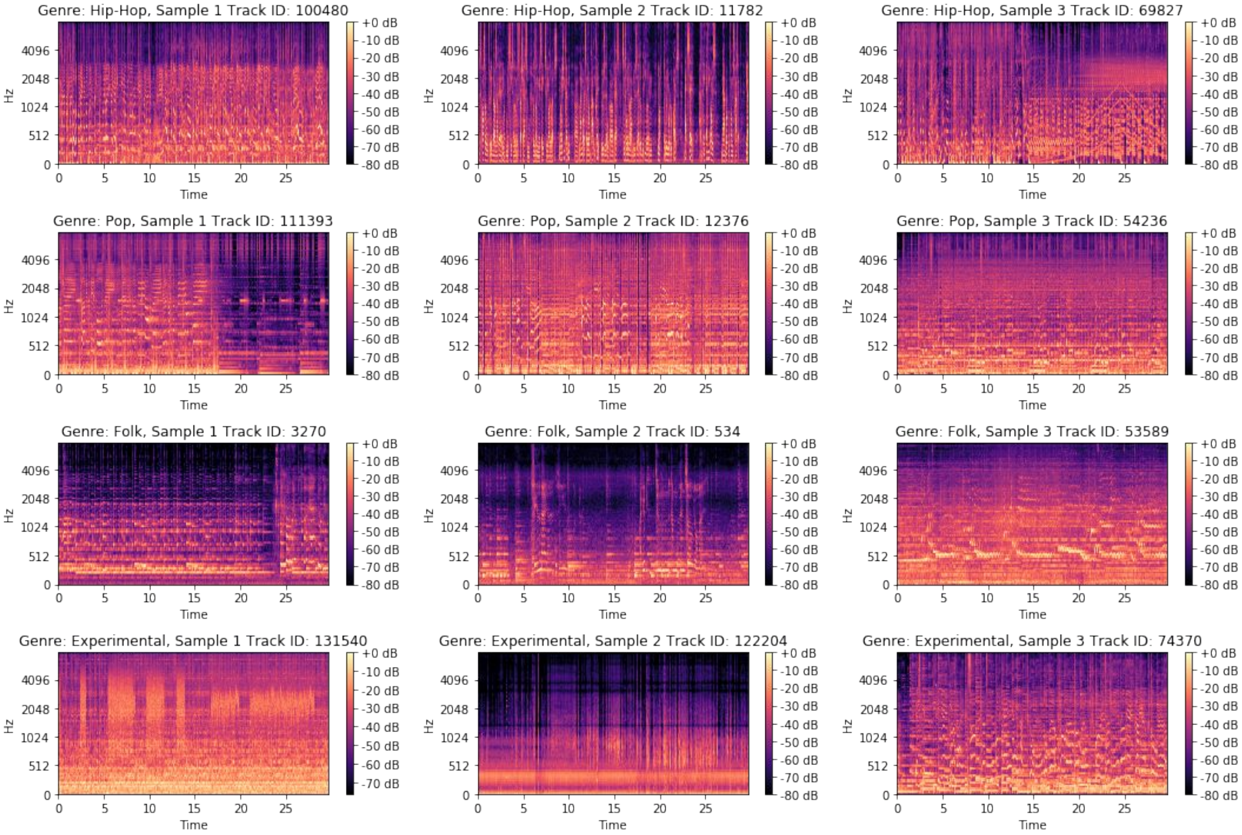
 # Human perception of sound intensity is logarithmic.
spect_db = librosa.power_to_db(S=spect, ref=np.max)

 # Plot mel spectrograms
librosa.display.specshow(spect_db, y_axis='mel',
fmax=8000, x_axis='time')

Mel** comes from **Melody
Psycho-acoustic scale of pitch



Example: Spectrograms



librosa Library Usage

```
# Other features that can be extracted

chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
rmse = librosa.feature.rmse(y=y)
spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
zcr = librosa.feature.zero_crossing_rate(y)
mfcc = librosa.feature.mfcc(y=y, sr=sr)
```

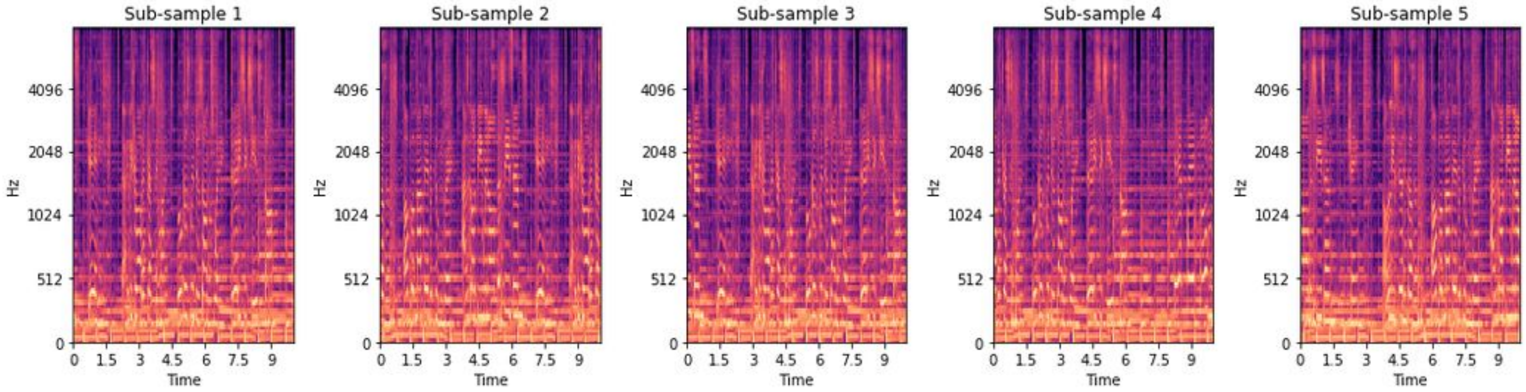
	filename	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	...
0	blues.00043.au	0.399025	0.127311	2155.654923	2372.403604	5012.019693	0.087165	-109.165355	100.621500	-8.614721	...
1	blues.00012.au	0.269320	0.119072	1361.045467	1567.804596	2739.625101	0.069124	-207.208080	132.799175	-15.438986	...
2	blues.00026.au	0.278484	0.076970	1198.607665	1573.308974	2478.376680	0.051988	-284.819504	108.785628	9.131956	...
3	blues.00077.au	0.408876	0.243217	2206.771246	2191.473506	4657.388504	0.111526	-29.010990	104.532914	-30.974207	...
4	blues.00084.au	0.396258	0.235238	2061.150735	2085.159448	4221.149475	0.113397	-38.965941	112.039843	-31.817035	...

Sub-Sampling the Data

Data: Free Music Archive, small - 8000 samples, balanced 8 genres.
Aim: Generate more data for training
Method: Randomly sampling shorter length samples from the 30s audio track

Example: 5 sub-samples of 10s from a single audio file

```
Genre: Folk
Sample file: ./data/fma_small/085/085486.mp3
Shape: (5, 128, 431)
```



2D Convolutional Neural Net

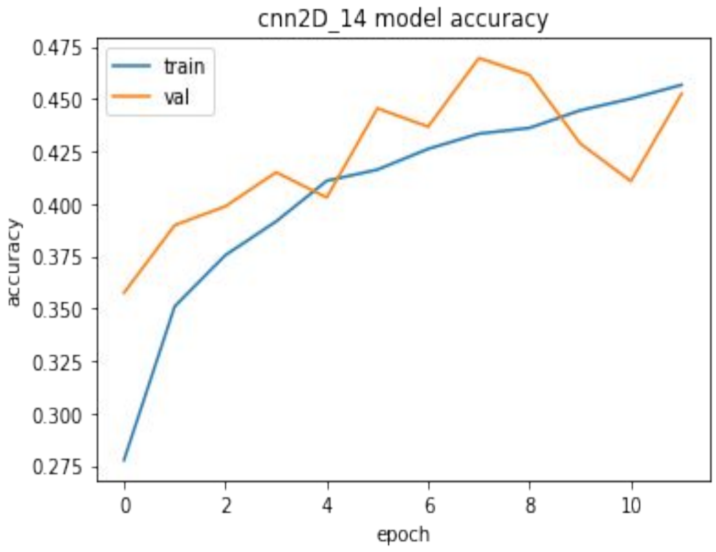
Base model: 2 x 2D Conv → 2 x Dense NN → Softmax Output layer
Input: 31,970 samples of training data, 431 x 128, single channel
Output: 8 possible genre classes

Directory of model parameters to be saved: ./models/cnn2D_14

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 216, 64, 8)	208
max_pooling2d_1 (MaxPooling2)	(None, 108, 32, 8)	0
conv2d_2 (Conv2D)	(None, 54, 16, 32)	2336
max_pooling2d_2 (MaxPooling2)	(None, 27, 8, 32)	0
flatten_1 (Flatten)	(None, 6912)	0
dense_1 (Dense)	(None, 32)	221216
dropout_1 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 8)	136

Total params: 224,424
Trainable params: 224,424

Max Val Accuracy: 0.46950



Hyperparameters

{'batch_size' : 32,
'kernel_size1' : (5,5),
'kernel_size2' : (3,3),
'filter_size1' : 8,
'filter_size2' : 32,
'strides1' : 2,
'strides2' : 2,
'padding' : 'same',
'activation' : 'relu',
'pool_size1' : 2,
'pool_size2' : 2,
'l2' : 0.01,
'epochs' : 12,
'optimizer' : 'adam',
'dense1' : 32,
'dense2' : 16,
'dropout1' : 0.2}

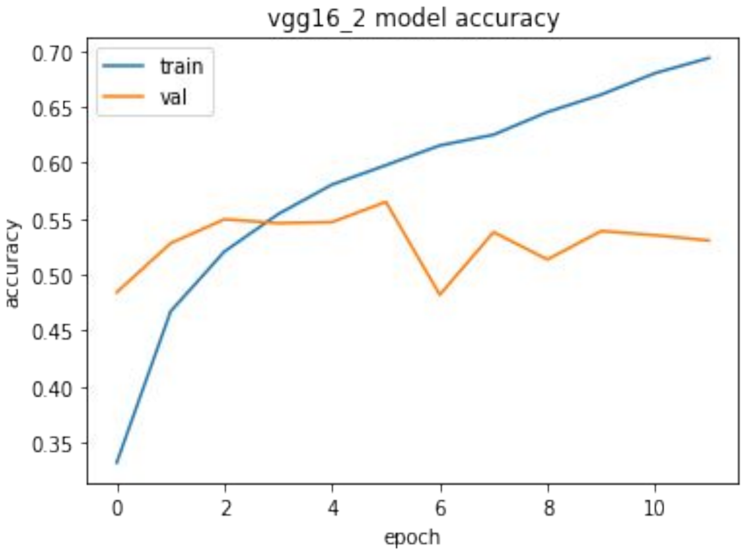
Transfer Learning with Pre-trained CNNs

Pre-trained model: VGG16 on ImageNet
Input: Conversion of input layer to accept single channel inputs from spectrograms

Layer (type)	Output Shape	Param #
spect_input (InputLayer)	(None, 431, 128, 1)	0
. *<blocks 1 - 4 are not displayed>*		
spect_block5_conv1 (Conv2D)	(None, 26, 8, 512)	2359808
spect_block5_conv2 (Conv2D)	(None, 26, 8, 512)	2359808
spect_block5_conv3 (Conv2D)	(None, 26, 8, 512)	2359808
spect_block5_pool (MaxPoolin	(None, 3, 1, 512)	0
flatten_1 (Flatten)	(None, 1536)	0
dense_1 (Dense)	(None, 64)	98368
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 16)	1040
dense_3 (Dense)	(None, 8)	136

Total params: 14,813,080
Trainable params: 7,178,968
Non-trainable params: 7,634,112

Max Val Accuracy: 0.5650



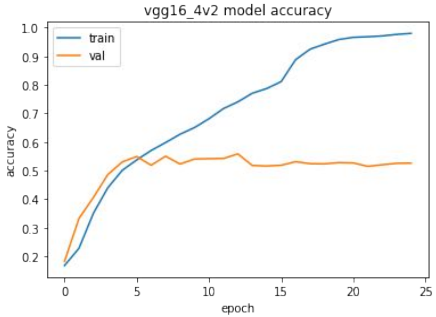
Hyperparameters

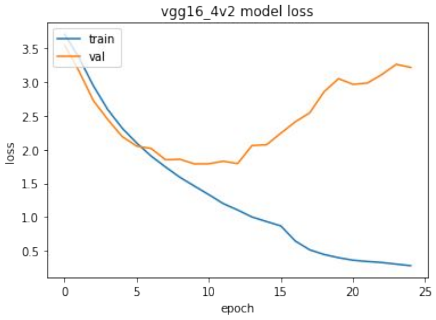
{'batch_size' : 32,
'non-trained_layers': 15,
'padding' : 'same',
'activation' : 'relu',
'l2' : 0.01,
'epochs' : 12,
'dense1' : 64,
'dense2' : 16,
'dropout1' : 0.5}

Transfer Learning with Pre-trained CNNs

VGG16

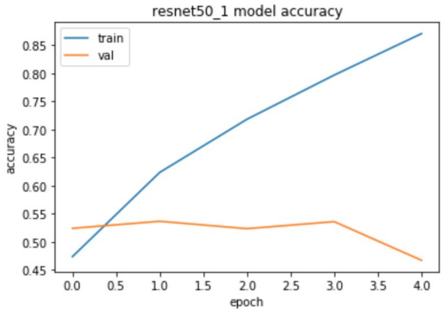
Accuracy: 0.56

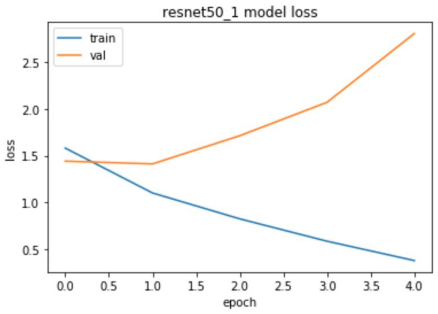




ResNet

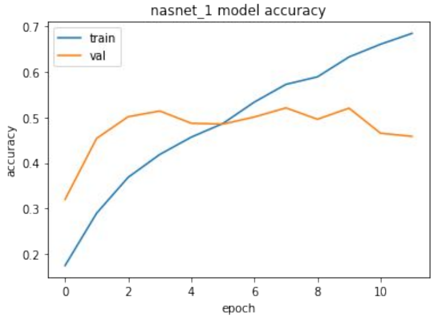
Accuracy: 0.53

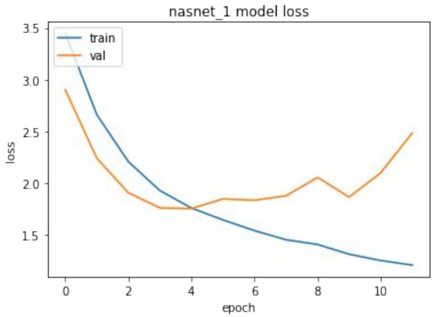




NASNet

Accuracy: 0.52



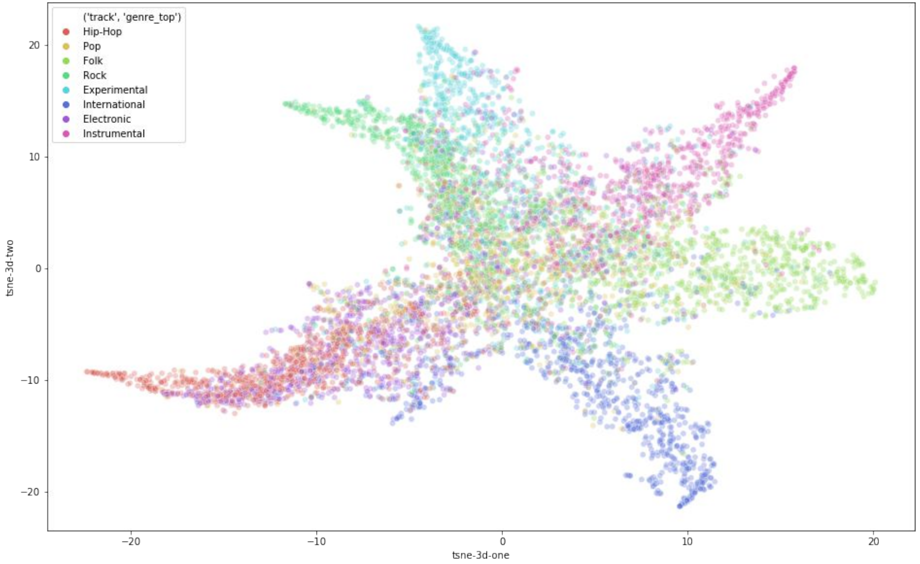


Transfer Learning Notes:

For these models, inputs were duplicated to match the 3 channel input required by the CNNs pre-trained on ImageNet.

Additional sets of subsampled data was fed into trained models for further fine-tuning, but no improvement in results.

Next Steps



- Song embeddings
- T-SNE
    - t-distributed stochastic neighbor embedding
- Web deliverable

Questions?

W210 Capstone Project, Week 10

Madeleine Bulkow | Kuangwei Huang | Weixing Sun